
TFG

Release 1.0

Hugo Ferreira

May 22, 2019

CONTENTS

1	Keywords documentation!	3
2	Grampal WS documentation!	5
3	Create_json documentation!	7
4	Elastic_bulk documentation!	9
	Python Module Index	11

Contents:

KEYWORDS DOCUMENTATION!

`keywords.Concatenate_candidates_grampal` (*graph*, *nodes*, *text*)
Get the multiwords from the top nodes of the graph using spacy as service.

Args: *graph* (*igraph*): Graph to be analyse.

nodes (*list*): The list of top nodes.

text (*str*): Text of origin.

Returns: *nodes* (*list*): The list of the multiwords.

`keywords.Concatenate_candidates_spacy` (*graph*, *nodes*, *text*)
Get the multiwords from the top nodes of the graph using spacy as service.

Args: *graph* (*igraph*): Graph to be analyse.

nodes (*list*): The list of top nodes.

text (*str*): Text of origin.

Returns: *nodes* (*list*): The list of the multiwords.

class `keywords.OrderedDict`
`OrderDict` class

Creates a new Dictionnary data structure that allows multiple append on the same key.

Args: *Dict* :The data structure Dictionnary.

`keywords.Pagerank` (*graph*)
Use the Google's pagerank algorithm to set a value for each node.

Args: *graph* (*igraph*): Graph to be analyse.

Returns: *values* (*list*): The list of values generated.

`keywords.Sort_occurences` (*graph*)
Get an array of the nodes sorted by occurence.

Args: *graph* (*igraph*): Graph to be analyse.

Returns: *nodes* (*list*): The list of nodes generated.

`keywords.Sort_values` (*graph*)
Get an array of the nodes sorted by value.

Args: *graph* (*igraph*): Graph to be analyse.

Returns: *nodes* (*list*): The list of nodes generated.

`keywords.Tnodes` (*graph*, *T*)
Extract the T nodes with higher values.

Args: graph (*igraph*): Graph to be analyse.

T (*int*): Number of nodes we want to get from the top.

Returns: nodes (*list*): The list of nodes generated.

keywords.**create_graph_grampal** (*text*, *k=2*)

Create a graph with the keywords and their links using grampal as service.

Args: text (*str*): The text of origin.

k (*int*): The correlation value ,by default = 2.

Returns: g (*igraph*): The graph generated.

keywords.**create_graph_spacy** (*text*, *k=2*)

Create a graph with the keywords and their links using spacy as service.

Args: text (*str*): The text of origin.

k (*int*): The correlation value ,by default = 2.

Returns: g (*igraph*): The graph generated.

keywords.**custom_tokenizer** (*nlp*)

Redefine the custom tokenizer of spacy .

Args: nlp (*nlp*): The tokenizer from spacy.

Returns: nlp (*nlp*): The new custom tokenizer.

keywords.**print_graph** (*graph*, *path*)

Print the graph generated, it was used for validation on small graph, currently unused .

Args: graph (*igraph*): The graph to be printed.

path (*str*): The path.

GRAMPAL WS DOCUMENTATION!

```
class ws.Grampal (service=None)  
    Grampal service class
```

This class implements all the functionality of the Grampal ws, allowing the tokenize and analyse of a phrase

analiza (*phrase*)

Analyse a phrase using Grampal's service

Args: phrase (*str*): The phrase to be analyse.

Returns: Object: The request object if successful, *None* otherwise.

The status_code of the response can be checked:

```
{ '200': 'success', '404': 'not found'  
}
```

analiza_get (*phrase*)

GET function of the Grampal service

Args: phrase (*str*): The phrase to be analyse.

Returns: Object: The request object if successful, *None* otherwise.

The status_code of the response can be checked:

```
{ '200': 'success', '404': 'not found'  
}
```

analiza_post (*phrase*)

POST function of the Grampal service

Args: phrase (*str*): The phrase to be analyse.

Returns: Object: The request object if successful, *None* otherwise.

The status_code of the response can be checked:

```
{ '200': 'success', '404': 'not found' }
```

info_lemma (*phrase*)

Parse the response from the Grampal ws extracting the lemma information

Args: phrase: Phrase to be analyse

Returns: String: The lemma information if successful, *None* otherwise.

info_orig (*phrase*)

Parse the response from the Grampal ws extracting the word of origin

Args: phrase: Phrase to be analyse

Returns: String: The word of origin of the token

info_syntactic (*phrase*)

Parse the response from the Grampal ws extracting the syntactic information

Args: phrase: Phrase to be analyse

Returns: String: The syntactic information if successful, *None* otherwise.

CREATE_JSON DOCUMENTATION!

`create_json.multiple_json(file_name)`

Function that creates multiple json (one for every row) from the babelnet index format

Args: `file_name: (str)`: The name of the index file.

`create_json.single_json(file_name)`

Function that creates a single json from the babelnet index format

Args: `file_name: (str)`: The name of the index file.

ELASTIC_BULK DOCUMENTATION!

`elastic_bulk.decode_nginx_log(_nginx_fd)`

Function that parse the source information from a json.

Args: `_nginx_fd (str)`: The name of the json file.

Returns: Object: The json object generated

`elastic_bulk.es_add_bulk(nginx_file)`

Function that bulk the information from a json.

Args: `nginx_file (str)`: The name of the json file.

PYTHON MODULE INDEX

C

`create_json` (*Unix, Windows*), [7](#)

E

`elastic_bulk` (*Unix, Windows*), [9](#)

K

`keywords` (*Unix, Windows*), [3](#)

W

`ws` (*Unix, Windows*), [5](#)

INDEX

`analiza()`ws.Grampal method, 5
`analiza_get()`ws.Grampal method, 5
`analiza_post()`ws.Grampal method, 5

`Concatenate_candidates_grampal()`in module keywords, 3
`Concatenate_candidates_spacy()`in module keywords, 3
`create_graph_grampal()`in module keywords, 4
`create_graph_spacy()`in module keywords, 4
`create_json`module, 7
`custom_tokenizer()`in module keywords, 4

`decode_nginx_log()`in module elastic_bulk, 9

`elastic_bulk`module, 9
`es_add_bulk()`in module elastic_bulk, 9

`Grampal`class in ws, 5

`info_lemma()`ws.Grampal method, 5
`info_orig()`ws.Grampal method, 5
`info_syntactic()`ws.Grampal method, 6

`keywords`module, 3

`multiple_json()`in module create_json, 7

`OrderedDict`class in keywords, 3

`Pagerank()`in module keywords, 3
`print_graph()`in module keywords, 4

`single_json()`in module create_json, 7
`Sort_occurences()`in module keywords, 3
`Sort_values()`in module keywords, 3

`Tnodes()`in module keywords, 3

`ws`module, 5